Dynamic multi-level appearance models and adaptive clustered decision trees for single target tracking

Xiao, Jingjing; Stolkin, Rustam; Leonardis, Ales

DOI:
10.1016/j.patcog.2017.04.001

Document Version
Peer reviewed version

Citation for published version (Harvard):
Xiao, J, Stolkin, R & Leonardis, A 2017, 'Dynamic multi-level appearance models and adaptive clustered decision trees for single target tracking', *Pattern Recognition*, vol. 69, pp. 169-183. https://doi.org/10.1016/j.patcog.2017.04.001

Dynamic multi-level appearance models and adaptive clustered decision trees for single target tracking

Jingjing Xiao, Rustam Stolkin, Aleš Leonardis

PII: S0031-3203(17)30148-6
DOI: 10.1016/j.patcog.2017.04.001
Reference: PR 6109

To appear in: *Pattern Recognition*

Received date: 9 October 2016
Revised date: 14 February 2017
Accepted date: 4 April 2017

Please cite this article as: Jingjing Xiao, Rustam Stolkin, Aleš Leonardis, Dynamic multi-level appearance models and adaptive clustered decision trees for single target tracking, *Pattern Recognition* (2017), doi: 10.1016/j.patcog.2017.04.001

Highlights

- The main contribution of the paper is an adaptive clustered decision tree approach which dynamically selects the minimum combination of features necessary to sufficiently represent each target part at each frame, thereby providing robustness without sacrificing computational efficiency.
- We show how this adaptive clustered decision tree can be utilised in two separate parts of the tracking algorithm: firstly to enable robust matching at the part level between successive frames; and secondly to select the best candidates (constructed form super-pixels) for learning new parts of the target.
- During matching, the adaptive clustered decision trees are used to search the set of candidates in the current frame, to find the best match to a target part in the previous frame. During model updating, the decision trees are used to search for the most suitable candidate to model a new part of the target, and to replace an old target part which drifts from the main body of the target in the current frame.

Dynamic multi-level appearance models and adaptive clustered decision trees for single target tracking

Jingjing Xiao[a,b], Rustam Stolkin[b], Aleš Leonardis[b]

[a]*Department of Medical Engineering, Xinqiao Hospital, Third Military Medical University, Chongqing 400037, China*
[b]*University of Birmingham, B15 2TT, U.K.*

Abstract

This paper presents a tracking algorithm for arbitrary objects in challenging video sequences. Targets are modelled at three different levels of granularity (pixel, parts and bounding box levels), which are cross-constrained to enable robust model relearning. The main contribution is an adaptive clustered decision tree method which dynamically selects the minimum combination of features necessary to sufficiently represent each target part at each frame, thereby providing robustness with computational efficiency. The adaptive clustered decision tree is used in two separate ways: firstly for parts level matching between successive frames; secondly to select the best candidate image regions for learning new parts of the target. We test the tracker using two different tracking benchmarks (VOT2013-2014 and CVPR2013 tracking challenges), based on two different test methodologies, and show it to be more robust than the state-of-the-art methods from both of those tracking challenges, while also offering competitive tracking precision. Additionally, we evaluate the contribution of each key component of the tracker to overall performance; test the sensitivity of the tracker under different initialization conditions; investigate the effect of using features in different orders within the decision trees; illustrate the flexibility of the method for handling arbitrary kinds of features, by showing how it easily extends to handle RGB-D data.

Keywords: Single target tracking, adaptive clustered decision trees, multi-level appearance models.

Email address: shine636363@sina.com (Jingjing Xiao)

1. Introduction

After several decades of visual tracking research, even the most sophisticated trackers are still prone to failure in challenging scenarios, including clutter and camouflage in one or more feature modalities, rapid and erratic target motion, occlusions, and targets which change their shape and appearance over time. These problematic tracking conditions predominantly lead to failures in three fundamental parts of the tracking algorithm: 1) the representation or model of the target object's visual appearance; 2) the mechanism for matching model parts to image regions at each frame; 3) the mechanism for continuously relearning or updating models of targets which change their appearance over time.

This paper presents a target tracking algorithm which achieves state-of-the-art robustness by addressing each of these three fundamental areas. We propose a flexible target representation which can adaptively exploit an arbitrary number of different image features. Targets are modelled at three different levels of granularity, including the level of individual pixels, the level of local parts (constructed from super-pixels), and a bounding box level which encodes overall information about the target as a whole. Cross-constraints between these different levels during updates enable continuous target model relearning which is robust and stable.

The main contribution of the paper is an adaptive clustered decision tree approach which dynamically selects the minimum combination of features necessary to sufficiently represent each target part at each frame, thereby providing robustness without sacrificing computational efficiency. We show how this adaptive clustered decision tree can be utilised in two separate parts of the tracking algorithm: firstly to enable robust matching at the part level between successive frames; and secondly to select the best candidates (constructed from super-pixels) for learning new parts of the target. During *matching*, the adaptive clustered decision trees are used to search the set of candidates in the current frame, to find the best match to a target part in the previous frame. During *model updating*, the decision trees are used to search for the most suitable candidate to model a new part of the target, and to replace an old target part which drifts from the main body of the target in the current frame.

We have carried out a principled evaluation using the latest benchmark methods, and comparing against the other state-of-the-art trackers. Results show that the proposed method outperforms the best trackers on both VOT2013 and VOT2014 benchmark sets. It outperforms the 7 available methods on the CVPR2013 dataset w.r.t. robustness, while also achieving competitive tracking accuracy. Furthermore, we have decomposed the tracker to evaluate the effectiveness of each com-

ponent, and evaluated the tracking performance with the various noisy initialization conditions. To have a deeper understanding of the proposed adaptive clustered decision trees, we also implemented the tracker on the publicly available RGB-D sequences and showed that, with well designed clustering methods, the tracker is relatively robust within various feature sequential orders.

The rest of the paper is organized as follows. Related work is discussed in Section 2. The multi-level target model, and its initialisation, is introduced in Section 3. Section 4 explains the adaptive clustered decision tree, and shows how it is used for both target matching and model updating at each successive frame. Section 5 presents and discusses the experimental results of testing the tracker on the VOT2013, VOT2014 and CVPR2013 benchmark video datasets. Additional experiments analyse the contributions of each key part of the tracker, to help explain the strong overall performance. Further more, we investigate how the noisy initialization affects the tracking performance. Section 6 shows how the decision trees can easily be extended to include arbitrary kinds of additional features, e.g., depth feature, illustrating how the decision trees dynamically vary the number of features exploited for each target part at each video frame. We further investigate the extent to which the tracker is robust to the order in which different features are represented at different tree levels. Section 7 provides concluding remarks and mentions ongoing efforts to extend this work in various new directions.

2. Related work

In this section, we review recent tracking algorithms in terms of three primary components: target representation, matching mechanism, and model update mechanism. We also discuss the relationship between our proposed adaptive clustered decision tree method and other kinds of tree-like, hierarchical or recursive classification methods from the literature.

2.1. Target representation

Choice of target representation is a crucial component of any tracker. Two main streams of research can be distinguished. The first uses holistic (overall) target templates for tracking, e.g., [27] [40]. However, such methods have difficulty in handling significant appearance changes and deformations of the target. Later work [18] [20], proposed patch-based approaches to provide more flexibility for target matching. However, the choice of geometric constraint for the local patches' movement remains an open problem, while environmental clutter can often distract such local patches and cause them to drift. [38] avoided

complex geometric constraints for patch motion, by treating the problem as a classification of foreground and background super-pixels. However, since each super-pixel is classified independently, this method remains prone to failure with cluttered background scenes. In contrast, our method also makes use of super-pixels, but exploits them within a more robust cross-constrained multi-level target model structure. More recent work [35] also combined both holistic and patch-based target models together for a more robust representation. However, this work fused multiple features in a homogeneous way (i.e., equally weighting the opinions of all feature modalities), which causes failures under conditions where one or more features become less discriminating than others. In contrast, our method achieves better results by adaptively selecting in favour of whichever feature or feature combination is most discriminating for each target part in each new frame.

2.2. Matching

To estimate the state of the target, the algorithm must match observations from a candidate image region to the target representation model. A single feature is not sufficient to handle large appearance variations, and recent work [29] [33] [23], increasingly exploits combinations of multiple features. One approach is to compute the likelihood from all features and then multiply all values to estimate the target state [35]. However, in such schemes, a poorly performing feature can degrade tracking performance, even when other features are highly discriminative. Therefore, instead of treating all features with equal importance, other methods, e.g. [8] [37] [33], attempt to identify and weight in favour of the most discriminative features at each time step. Brasnett et al. [6], propose a scheme for weighting in favour of the best performing features, and updating these weights adaptively at each new frame. However, this method ignores feature saliency from the local background regions. In contrast, recent work [33] proposes an adaptive method which successfully exploits contextual information for optimally weighting the contributions from each feature online during tracking. However, [33] uses only a simple holistic target model which is insufficient to cope with large target deformations and appearance changes. Pernici et al. [29] propose a matching method which uses both the target and context SIFT features. However, the matching indices are obtained directly by a nearest neighbour search, which might perform poorly when the target undergoes rapid and significant deformations and appearance changes. In contrast to the homogeneous treatment of all features by e.g. [35], our adaptive clustered decision tree approach can adaptively select in favour of the most discriminative features for matching each target part to each new frame. Moreover, this adaptive feature selection is also embedded within

a cross-constrained multi-level target representation, which enables much more robust matching and model updating than the simple holistic target models of e.g. [33].

2.3. Model update mechanisms

For robust, general tracking, it is essential to continuously update or relearn the target model to cope with appearance changes. An appropriate target model should enable the tracker to overcome errors in the relearning process which might corrupt the target model, and support long term tracking without drifting [24]. Early methods, such as [27], updated the model at every frame as a simple linear combination of the previous model and the most recent estimation of the target region in the current image. Without additional methods for precise delineation of the target parts, such update methods are likely to fail, given sufficiently long tracking duration, due to accumulated errors and noise during successive updates. In MIL [4], and other trackers such as OB [12] and SB [13], updating of the target model is performed by an evolving boosting classifier that tracks image patches and learns the object appearance. However, online boosting requires that the data should be independent and identically distributed, which is a condition not satisfied in most real video sequences, where data is often temporally correlated [29]. A more robust updating mechanism is achieved by [35], which forms a cross-constraint paradigm to stably constrain the relearning of a two-layer target model, in which global (bounding region) and local (parts based) models are used to constrain (and thereby stabilise) each others' online relearning. However, this method (and most earlier methods e.g. [27]) updates target appearance models at a fixed rate, regardless of the confidence (or lack of) in current target observations. This problem is compounded by the previously described problem, that many methods, e.g. [35], combine the opinion of all features with equal weight, which can lead to drifting of patches away from the true target location when one or more feature modalities are poorly discriminative compared to others. Like [35], we also employ a multi-level cross-constraint approach to robustify online target relearning, however we achieve improved performance evaluation results over [35] by adaptively fusing multiple features in the proposed decision trees and varying the relearning rate, for each target part at each frame, based on a current tracking confidence measure.

2.4. Boosting, decision trees and recursive classification structures

It is possible to draw some (very loose) analogies between boosting and our proposed adaptive decision tree structure. Both concepts are, in some sense, aimed

at successively combining elements of different features (or classifiers) to enable a more robust overall classification. Adaptive Boosting [11] involves starting with a single weak classifier, and recursively adding the opinions of additional weak classifiers, with confidence weights for each successive classifier being chosen to minimise the overall error with respect to ground-truthed training data. Analogously, our proposed adaptive clustered decision tree begins with a single feature, and recursively adds further features until a robust decision can be achieved. However, this analogy is limited: the original Adaboost uses a weighted combination of all weak classifiers, whereas the adaptive clustered decision tree adaptively incorporates only those features necessary to make a decision with the required degree of confidence; Adaboost relies on a set of training data with known ground-truth whereas our adaptive clustered decision tree is incorporated into a system for continuous online target matching and relearning with no such ground truth available; Adaboost is a (typically binary) classifier, whereas the proposed adaptive decision tree is used to search for the best possible out of an arbitrary number of candidates in order to achieve matching or model updating. More relevant to the present work, Adaboost has been applied to the problem of visual object tracking. Notably, [13] and [28] use individual bins of a histogram (which could be colour or another feature) as "weak classifiers" and employ Adaboost as a way of creating a weighted combination of these classifiers which best discriminates between foreground and surrounding background regions at each frame. These methods use a linear combination of all available features at each frame, whereas our proposed adaptive clustered decision tree uses a minimum feature set, only adding additional features when the previous features are insufficient to achieve robust matching. In other words, the computational burden of our method depends on, and varies with, the visual complexity of the scene in each successive video frame. Furthermore, each successive level of the adaptive decision tree, does not utilise weak classifiers, but relies on strong classifiers, i.e., the full histogram or distribution for that layer's feature modality. Adaboost is often combined with decision tree classifiers. Probabilistic Boosting Trees [34] have been used for classification, recognition and clustering problems. Each node of the tree combines weaker classifiers (lower nodes), yielding a top node which outputs an overall posterior probability by integrating the conditional probabilities gathered from its sub-trees. However, this method is less well suited to online target tracking problems, since the tree structure remains fixed after its initial training. In contrast, our proposed adaptive decision tree is relearned for each target part at each frame, and enables robust online adaptation to targets which continuously change their appearance with time, as well as background scenes which change their complexity, clutter

and camouflage properties with time. Somewhat related to our proposed adaptive clustered decision tree structure [26] proposes a vocabulary tree for classification by hierarchically quantising features. The method is applied to computing a relevance score between a query image and database images. Initially, k-means clustering is applied to the training data, partitioning it into k quantisation cells. This process is recursively applied to each group of descriptor vectors to define a tree level by level. A query image can now be classified by comparing its path through the tree, with that of images from a database. In contrast to this approach, which forms k feature clusters on each tree level, our proposed adaptive decision tree structure clusters by recursively comparing the similarity score between candidates, explicitly with a defined context-based similarity threshold method which is designed to provide robustness to dynamically changing scenes. Other recent work [23] simultaneously tracks, learns and parses targets using a hierarchical and compositional And-Or graph (AOG) representation. Nodes in the AOG tree-like structure are intended to represent the local parts of the target object. In contrast, the nodes in our adaptive clustering tree are different features which are used for one specific part matching. Therefore, theoretically, our proposed tree can be utilized for the node matching in AOG tree. Additionally, the AOG algorithm of [23] uses a fixed cell grid to quantize the AOG structure, which therefore may contain nodes that do not necessarily correspond to consistent or distinct target parts. In contrast, our method links tree nodes directly to target parts, which are more likely to represent homogeneous and meaningful parts of the target object. More recent work is proposed in [17] which conducted tracking based on a tree-structured target appearance model. A number of key frames are used, with a semi-supervised manifold alignment technique, to estimate an optimal tree which is applied for tracking all remaining frames. However, this off-line tracker is less suitable for online tracking of targets which continuously change their shape and visual appearance over time.

3. Multi-level target representation

In the spirit of [2] and [39], our method initialises its target representation using only a given bounding box in the first frame. The target is modelled hierarchically at three different levels of granularity: the pixel (bottom) level, part (constructed from super-pixel) level, and the target (top or overall representation) level, see Fig. 1. Following the logic of the model construction and the order of model updating, this section first introduces the middle level model, since the initial features are extracted from this level. Later, we introduce the top level

followed by the bottom level. We precede the model descriptions with a brief overview of notation.

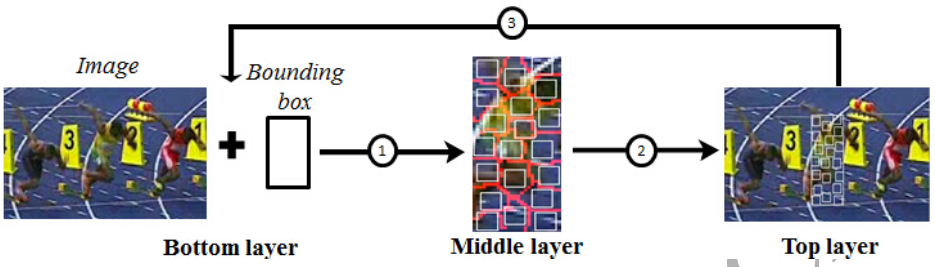

Figure 1: 1-use top level propagation to find a candidate image region, and segment it into super-pixels; 2-match middle level parts, update old parts, learn new parts, use updated parts to relearn top level; 3-use new top level to assign likelihoods to bottom level pixels in the new frame.

3.1. Notation

Our method can be used with any combination of features:

$$\mathbf{F}_k = \{\mathbf{f}_k^1, ..., \mathbf{f}_k^i, ..., \mathbf{f}_k^{N_f}\} \tag{1}$$

where $\mathbf{F}_k$ denotes a set of N_f feature modalities exploited by the tracker at the k^{th} video frame, and $\mathbf{f}_k^i$ denotes a particular kind of feature, provided that all such features, $\mathbf{f}_k^i$, satisfy the following conditions.

Firstly, it should be possible to associate any image pixel with a feature value:

$$\mathbf{I}_k(x) \mapsto \mathbf{f}_k^{i(x)} \tag{2}$$

where $\mathbf{I}_k(x)$ denotes the pixel at location x in the k^{th} image, $\mathbf{I}_k$. Clearly, certain kinds of features, e.g. texture, must be calculated from a region surrounding an individual pixel, but such feature values can still be associated with the central pixel's location.

Secondly, in the i^{th} feature modality, the set of feature values $\mathbf{f}_k^{i(x \in \boldsymbol{R}_k^q)}$ of any image region, $\boldsymbol{R}_k^q$, can be used to create a model:

$$\boldsymbol{R}_k^q \mapsto \boldsymbol{M}_k^{i(q)} \equiv \boldsymbol{M}_k^{i(x \in \boldsymbol{R}_k^q)} \tag{3}$$

where $\boldsymbol{M}_k^{i(q)}$ is a model of the q^{th} region's pixels' feature values, which could for example be a probability distribution or some other kind of collective representation.

Often, we make use of a q^{th} image region, $\boldsymbol{R}_k^q$, which corresponds to a q^{th} super-pixel, $\boldsymbol{S}_k^q$. In such cases, the super-pixel will yield N_f feature models in the N_f feature modalities:

$$\boldsymbol{S}_k^q \mapsto \{\boldsymbol{M}_k^{1(q)} ... \boldsymbol{M}_k^{i(q)} ... \boldsymbol{M}_k^{N_f(q)}\} \tag{4}$$

where $\boldsymbol{S}_k^q$ is the q^{th} out of a set of N_s super-pixels at frame k.

Thirdly, for the i^{th} feature modality, a suitable likelihood function $\hat{L}_k^{i(p,q)}$ exists for comparing the similarity of any pair of models, $\boldsymbol{M}_k^{i(p)}, \boldsymbol{M}_k^{i(q)}$, representing any pair of image regions, $\boldsymbol{R}_k^p$, $\boldsymbol{R}_k^q$:

$$\{\boldsymbol{M}_k^{i(p)}, \boldsymbol{M}_k^{i(q)}\} \mapsto \hat{L}_k^{i(p,q)} \tag{5}$$

Where such feature models are in the form of distributions, then corresponding likelihood functions are typically based on divergence measures, such as [5], [19], [22], [31]. However, in general, other kinds of useful likelihood functions can be formed for other kinds of models.

Fourthly, for each kind of feature model $\boldsymbol{M}_k^{i(q)}$, another likelihood function $\tilde{L}^{i(x,q)}$ exists for evaluating the consistency of any pixel's feature value, $\mathbf{f}_k^{i(x)}$, with the model. Intuitively, we can regard the set of such likelihoods:

$$\tilde{\boldsymbol{L}}_k^{(x,q)} = \{\tilde{L}_k^{1(x,q)} ... \tilde{L}_k^{i(x,q)} ... \tilde{L}_k^{N_f(x,q)}\} \tag{6}$$

as the set of opinions of each feature modality as to the likelihood of such a pixel value belonging to the part of the target object projected as the q^{th} image region, $\boldsymbol{R}_k^q$, in the k^{th} frame. In cases where the feature models are in the form of distributions, then such likelihoods may be conveniently posed in the form of conditional probabilities.

3.2. Middle level target representation

The middle level consists of a set of feature models (in the sense of Eq. 3), extracted from the image regions corresponding to parts of the target, which we detect as super-pixels (Eq. 4). Using super-pixels as the basis for middle level model extraction offers several advantages over randomly selecting square "patches" as in previous methods [35] [41]. Firstly, a super-pixel is much more likely to correspond to a distinct and homogeneous part of the target. In contrast, randomly (or uniformly) selected patches are likely to contain pixels from two or more dissimilar (e.g. in terms of colour) parts of the target, which can lead to matching ambiguity. Secondly, when patches are randomly (or uniformly) selected from an

initial bounding box, many patches are likely to contain information about both target and background pixels, and this also can negatively impact tracking performance. In contrast, due to the homogeneous nature of super-pixels, the pixel models extracted from these regions are much more likely to include either purely target pixels, or purely background pixels. Once tracking begins, those super-pixels which erroneously correspond to background regions are rapidly detected and eliminated from the middle level target representation, leaving only those super-pixels which truly belong to target parts. We use the SLIC super-pixel segmentation method [3], because it offers the following advantages: the number of super-pixels can be known in advance; the super-pixels have uniform size; the compactness can be defined; and the algorithm has high computational efficiency. Then, the middle level target representation, $\zeta_k^{\texttt{mid}}$, consists of $N_{\texttt{mid}}$ feature models (of the form Eq. 3), for each of N_s super-pixels, representing N_s target parts, denoted as:

$$\begin{aligned}\zeta_k^{\texttt{mid}} = \{\{\boldsymbol{M}_k^{1_{\texttt{mid}}(1)} ... \ \boldsymbol{M}_k^{i_{\texttt{mid}}(1)} ... \ \boldsymbol{M}_k^{N_{\texttt{mid}}(1)}\},\\ ..., \{\boldsymbol{M}_k^{1_{\texttt{mid}}(q)} ... \ \boldsymbol{M}_k^{i_{\texttt{mid}}(q)} ... \ \boldsymbol{M}_k^{N_{\texttt{mid}}(q)}\},\\ ..., \{\boldsymbol{M}_k^{1_{\texttt{mid}}(N_s)} ... \ \boldsymbol{M}_k^{i_{\texttt{mid}}(N_s)} ... \ \boldsymbol{M}_k^{N_{\texttt{mid}}(N_s)}\}\}\end{aligned}$$

where $\boldsymbol{M}_k^{i_{\texttt{mid}}(q)}$ is a feature model in the i^{th} feature modality for the q^{th} super-pixel, representing the q^{th} target part, at frame k.

3.3. Top level target representation

The top level of the target representation is denoted as $\zeta_k^{\texttt{top}}$ which includes overall information about the target's bounding box (foreground) region, $\boldsymbol{R}_k^F$, and also a ring-shaped local background region, $\boldsymbol{R}_k^B$, which surrounds the foreground region (see Fig. 2), in the form of a set of models for each region in each feature modality:

$$\zeta_k^{\texttt{top}} = \{\boldsymbol{M}_k^{i_{\texttt{top}}(F)}, \boldsymbol{M}_k^{i_{\texttt{top}}(B)}\}_{i=1...N} \tag{7}$$

The top level target models, $\{\boldsymbol{M}_k^{1_{\texttt{top}}(F)} ... \ \boldsymbol{M}_k^{i_{\texttt{top}}(F)} ... \ \boldsymbol{M}_k^{N_{\texttt{top}}(F)}\}$, are computed from the middle level target parts models as:

$$\boldsymbol{M}_k^{i_{\texttt{top}}(F)} = \frac{1}{N_s} \sum_{q=1}^{N_s} \boldsymbol{M}_k^{i_{\texttt{mid}}(q)} \tag{8}$$

where $\{\boldsymbol{M}_k^{i_{\text{mid}}(q)}\}_{q=1...N_s}$ are the middle level feature models for each of N_s target parts (super-pixels), as defined in Eq. 7. In contrast, the top level local background models, $\{\boldsymbol{M}_k^{1_{\text{top}}(B)}...\ \boldsymbol{M}_k^{i_{\text{top}}(B)}...\ \boldsymbol{M}_k^{N_{\text{top}}(B)}\}$, are computed directly from the pixels occupying the ring-shaped local background region surrounding the target:

$$M_k^{i_{\text{top}}(B)} \equiv M_k^{i_{\text{top}}(x \in \boldsymbol{R}_k^B)} \tag{9}$$

where $\boldsymbol{R}_k^B$ is determined by enlarging the target's bounding box area by multiplying a scale factor λ (value 1.2) to both width and height, as shown in Fig. 2.

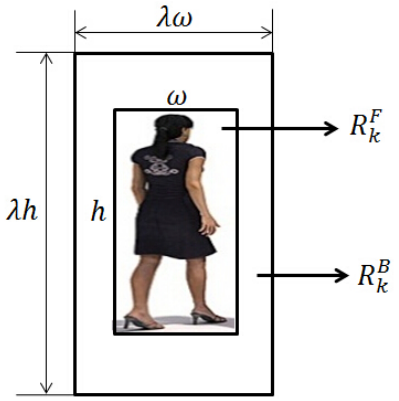

Figure 2: Foreground, $\boldsymbol{R}_k^F$, and ring-shaped local background, $\boldsymbol{R}_k^B$, regions. h and ω is the height and width of the bounding box, respectively. The scaling factor is λ.

3.4. Bottom level target representation

The bottom level representation, ζ_k^{bot}, comprises individual pixels, each associated with a set of likelihoods, according to each feature modality, that the pixel belongs to the target object:

$$\zeta_k^{\text{bot}} = \{l_k^{1(x,\text{bot})}..., l_k^{i(x,\text{bot})}..., l_k^{N(x,\text{bot})}\}_{x \in (\boldsymbol{R}_k^F \cup \boldsymbol{R}_k^B)} \tag{10}$$

where $l_k^{i(x,\text{bot})}$ denotes the discriminative likelihood of a pixel belonging to the target, computed as:

$$l_k^{i(x,\text{bot})} = \frac{\lambda^{\text{bot}} \tilde{L}_k^{i(x,F)}}{\lambda^{\text{bot}} \tilde{L}_k^{i(x,F)} + (1-\lambda^{\text{bot}}) \tilde{L}_k^{i(x,B)}} \tag{11}$$

where, with respect to the i^{th} feature modality: $\tilde{L}_k^{i(x,F)}$ denotes the likelihood of a pixel value, $\mathbf{I}_k(x)$, with respect to the top level model, $\boldsymbol{M}_k^{i_{\mathtt{top}}(F)}$, of the target foreground region, $\boldsymbol{R}_k^F$; $\tilde{L}_k^{i(x,B)}$ denotes the likelihood of the pixel value with respect to the top level model, $\boldsymbol{M}_k^{i_{\mathtt{top}}(B)}$, of the target's local background region, $\boldsymbol{R}_k^B$; $\lambda^{\mathtt{bot}}$ is the expected ratio of top level foreground area to the top level background area. For simplification, it can be computed as : $\lambda^{\mathtt{bot}} = (\lambda - 1)/2$.

4. Tracker propagation and matching

An overall schematic for the tracker is shown in Fig. 3. The tracker is first propagated by information matching at the top level, which generates a candidate image region for the middle level. Next, this candidate region is segmented [3], into equally sized super-pixels. We next propose a continuously-adaptive clustered tree method, which efficiently associates middle level target parts (learned from super-pixels in the previous frame), onto candidate super-pixels in the new frame, using the minimum number of features for each part. Lastly, to cope with target deformation and appearance changes, well matched parts are adaptively updated, while parts which cannot be satisfactorily matched are deemed to no longer be visible (e.g. due to occlusion) and are temporarily "switched off", but stored in memory, from the mid-level model. The severely drifting parts, defined as those drifting far from the main body of the target, will be replaced by new candidates. In this case, another decision tree is then used to choose the best super-pixels for creating new middle level parts to replace those drifting parts.

4.1. Top level propagation

The top level is propagated using a set of uniformly distributed samples $\{\hat{R}_k^{\mathtt{top}}(j)\}_{j=1...N_R}$ to represent the posterior density function of the target, with associated weights.

$$\hat{R}_k^{\mathtt{top}}(j) = R_{k-1}^{\mathtt{top}} + V_k(j)^{\mathtt{top}} \tag{12}$$

where $\hat{R}_k^{\mathtt{top}}(j)$ represents the hypothetical state (region) of the target computed from the previous tracker state $R_{k-1}^{\mathtt{top}}$ with a uniformly assigned moving velocity $V_k(j)^{\mathtt{top}}$.

Conventional methods, e.g. [27], estimate the overall target position using the expectation operator over the set of samples, which can be denoted as:

$$R_k^{\mathtt{top}} = \sum_{j=1}^{N_R} \hat{R}_k^{\mathtt{top}}(j) p(\hat{R}_k^{\mathtt{top}}(j) | R_{k-1}^{\mathtt{top}}) \tag{13}$$

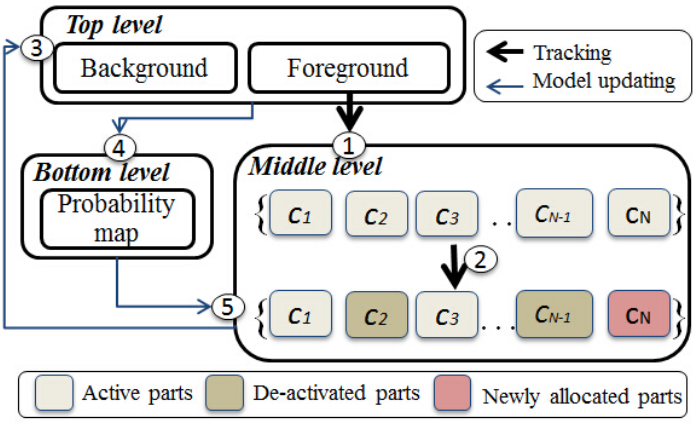

Figure 3: Block diagram of the tracking process. 1- tracker propagation by foreground information matching at the top level; 2- middle (super-pixel) level matching with clustered decision tree; 3- update the top level model; 4- feed back the bottom (pixel) level information; 5- re-sample drifting parts at the middle level.

where $\hat{R}_k^{\text{top}}(j)$ represents the hypothetical state (region) of the target, referred to j^{th} sample at k frame, with associated normalized weight $p(\hat{R}_k^{\text{top}}(j)|R_{k-1}^{\text{top}})$ such that $\sum_{j=1}^{N_R} p(\hat{R}_k^{\text{top}}(j)|R_{k-1}^{\text{top}}) = 1$ and N_R is the number of the samples. Here, the associated weights of the underlying samples are computed from the similarity between the target top foreground models and the candidate regions' models. For one specific model i of the sample j, the corresponding weight can be denoted as:

$$p(\hat{R}_k^{i_{\text{top}}}(j)|R_{k-1}^{i_{\text{top}}}) = exp\{\frac{\hat{L}_k^{i_{\text{top}}}(\hat{M}_k^{i_{\text{top}}(F)}(j), M_{k-1}^{i_{\text{top}}(F)})^2 - 1}{\sigma_{i_{\text{top}}}^2}\} \tag{14}$$

where $\hat{M}_k^{i_{\text{top}}(F)}(j)$ is the i^{th} foreground model of sample j in top layer, while $M_{k-1}^{i_{\text{top}}(F)}$ is the corresponding reference model. $\sigma_{i_{\text{top}}}$ is the standard deviation and $\hat{L}_k^{i_{\text{top}}}$ is the same likelihood function in Eq. 5. Our later proposed adaptively clustered decision trees could be used for fusing the weights computed from different models of the same sample.

4.2. Middle level matching with adaptive clustered decision tree

After propagating the top level to give a candidate bounding box region, this is further enlarged to form a broader search region, which we then segment into super-pixels [3], which provide a pool of candidate image regions for matching to middle level target parts, using our proposed adaptive clustered decision

tree method. Typically, the structures of decision trees are generated offline during training, e.g. [17] [34], but such static trees would be unsuitable for tracking targets with dynamically changing appearances, against changing background scenes. Instead, we propose an adaptive tree structure, which is relearned online for each new image, by explicitly considering contextual information.

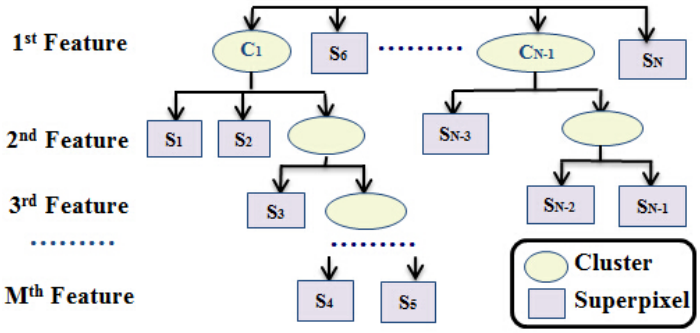

Figure 4: Clustered decision tree. Each tree-level represents a feature modality (e.g. color, motion etc.). The feature values of superpixels form leaves on a tree-level. If any two leaves are sufficiently similar in feature value then they are merged into a cluster. A new tree-level (using the next feature modality) is then used to try to disambiguate the members of such clusters. The tree continues growing (by adding more tree-levels of more features), until all target parts have been assigned to a unique choice of superpixel. Any remaining unmatched parts are assumed to have become occluded and are temporarily switched off.

The proposed adaptive clustered decision tree method is illustrated in Fig. 4. The objective of the decision tree is to find the corresponding super-pixel which best matches each middle level target part, while adaptively selecting the minimum combination of features needed to do this robustly in each frame. The first tree-level is initialised by selecting a primary feature from the set of all features, and labelling all candidates (constructed from super-pixels) as individual leaves. Next, all leaves are compared for similarity in the primary feature modality. Similar candidates are grouped into clusters according to a label map:

$$\mathfrak{C}_k^{i_{\text{mid}}(p_s,q_s)} = \begin{cases} 1, & \hat{L}_k^{i_{\text{mid}}(p_s,q_s)} > T_k^{i_{\text{mid}}} \\ 0, & others \end{cases} \tag{15}$$

where $\hat{L}_k^{i_{\text{mid}}(p_s,q_s)}$ is the similarity metric for i^{th} feature model between candidates p_s and q_s and $T_k^{i_{\text{mid}}}$ is a clustering threshold (relearned online as described later).

Next, each terminating leaf and cluster is compared against the part in the middle level target model for which a match is sought.

$$\arg\max_{(p_s,q_t)} \hat{L}_k^{i_{mid}(p_s,q_t)} \quad s.t. \;\; \hat{L}_k^{i_{mid}(p_s,q_t)} > 0 \tag{16}$$

where q_t indicates the target part, p_s is the candidate leaf or cluster, and $\hat{L}_k^{i_{mid}}$ is the similarity metric of feature model i_{mid}.

If any candidate super-pixel is both i) distinct enough from others that it forms its own leaf and does not lie inside a cluster (e.g. S_6 or S_N in Fig. 4) and ii) strongly matches the target part, then the decision tree ceases splitting and the middle level target part is labeled as matching that candidate local region. However, if the best match p_s for middle level part q_t is one of the clusters, then this suggests that the primary feature is not sufficiently discriminating to enable q_t to be matched to a unique super-pixel. We therefore grow a new tree-level, based on a second feature, i.e. the cluster grows a new child leaf for each of its constituent candidates (e.g. S_1 or S_2 on second row in Fig. 4), where each leaf in the new tree-level is modelled using the secondary feature modality. This tree extending procedure continues (by adding additional tree-levels for additional features) until a unique patch-to-leaf correspondence is found, e.g. C_1 or C_{N-1} in Fig. 4. Once all the middle level parts have been matched to new locations in the current image, the boundary of their distribution is used to output a new bounding box position, and the top level target models are updated accordingly.

Occasionally, some parts in the model will fail to find a strongly matching candidate, even after exhausting all possible image features (corresponding to all possible tree levels). In such cases, it is inferred that the part is no longer visible, and will be switched off in the model adaptation stage (it will again be activated during the matching process in the next frame). Other methods in the literature (e.g. [35]) remove unmatched patches, and thus lose parts of the model during occlusions, which cannot later be recovered. In contrast, our proposed approach of temporarily switching off the unmatched parts provides a powerful tracking memory that automatically handles occlusion situations without the need for special additional occlusion routines.

Note that the threshold, $T_k^{i_{mid}}$, has an important and useful effect on the extent to which each feature is used in the mid-level patch matching procedure. High values of $T_k^{i_{mid}}$ reduce the amount of clustering on the respective tree-level, ensuring that most candidate local regions will be represented as individual leaves, i.e. the algorithm will distinguish most candidates using this feature alone. In contrast, low values of $T_k^{i_{mid}}$ make it likely that this tree-level will grow many clusters, with the effect that the overall algorithm will make less use of this particular feature for matching, and will rely on later features (lower layers on the

tree) to provide discrimination. In other words, the choice of $T_k^{i_{mid}}$ can actually be regarded as a measure of confidence in the discriminating ability of a feature.

Consequently, we would like to set high values of $T_k^{i_{mid}}$ for those features that are highly discriminatory in the current frame, and low values of $T_k^{i_{mid}}$ for poorly performing features (e.g. those modalities for which the target is camouflaged against the background in the present frame). We therefore propose a method by which $T_k^{i_{mid}}$ is continuously relearned for each feature at each frame, based on evaluating the feature's discriminating ability relative to the current contextual information. $T_k^{i_{mid}}$ is computed online as:

$$T_k^{i_{mid}} = \hat{L}_k^{i_{top}(F,B)} \tag{17}$$

where $\hat{L}_k^{i_{top}(F,B)}$ indicates the similarity metric of models $\boldsymbol{M}_k^{i_{top}(F)}$, $\boldsymbol{M}_k^{i_{top}(B)}$ which are the top level target and ring-shape models defined in Eq. 7, and Eq. 17 is therefore a measure of similarity between the target and the local background (i.e. a measure of camouflage) in the respective feature modality.

The advantages of the adaptive decision tree are: firstly, its structure is adaptively generated online, which overcomes the over-fitting of offline generated classifiers; secondly, rather than using all features in all frames, the adaptive tree efficiently exploits only the minimum number of necessary features, ensuring sufficient robustness for minimal computational cost; thirdly, the decision tree adaptively weights in favour of the most discriminating features, dynamically responding to changing amounts of camouflage in different feature modalities.

4.3. Model updating

For robust tracking, it is necessary to continuously update the target model as it changes its appearance with time. The proposed tracker does this via two mechanisms: adapting the old middle and top level target models, and adding new models of new middle level target parts (derived from super-pixels).

4.3.1. Learning new target parts

At each frame, we examine all the matched middle level target part models and detect those which are drifting (moving too far from the target centroid), with a method adapted from our recent work [41]. The algorithm first searches the maximum non-zero interval of the marginal distribution according to the matched parts. The region outside the maximum non-zero interval becomes regarded as a potential drifting region and is then chosen as a candidate for additional drift checking according to: 1. only a small group of parts, which are separated from

the majority cluster; 2. the distribution density of the drifting patches should be very small.

To replace those middle level target parts that were detected as drifting, we select the "available" candidate local regions, i.e. those which are not yet matched to a target part, and rank them in order of their likelihood of representing target parts. A second kind of adaptive clustered decision tree is used to perform this ranking as follows.

We begin by using the primary feature modality to generate the leaves of the first level of the decision tree, according to a ranked (descending order) list of unmatched candidate likelihoods $\mathbf{S}_k^{i_{mid}(p_s)}$ (explained later in the experiment section). We then cluster those candidates (super-pixels) which share similar likelihoods, according to:

$$\mathfrak{C}_k^{i_{mid}(p_s,q_s)} = \begin{cases} 1, \|\mathbf{S}_k^{i_{mid}(p_s)} - \mathbf{S}_k^{i_{mid}(q_s)}\| < \lambda_{rank}\sigma_{all} \\ 0, \quad others \end{cases} \tag{18}$$

where λ_{rank} is a pre-defined parameter while σ_{all} is the standard deviation of all candidates' expected likelihoods. p_s and q_s is the index of candidate local regions.

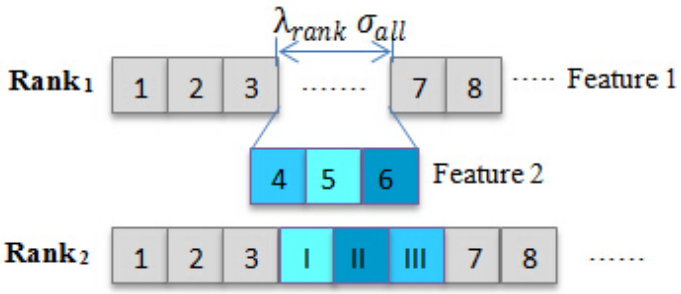

Figure 5: Rank the candidate local regions with clustered decision trees (descending order). $Rank_1$ ranks the candidates only using primary feature; $Rank_2$ re-ranks the candidates by adding additional feature.

Once again, a cluster suggests that the feature for this tree-level is not sufficiently discriminative to achieve a robust ranking. Therefore, a secondary feature is chosen and used to rank all constituent candidates within the cluster, forming a second tree-level.

The tree is grown (by adding successive tree-levels, using successive features), until a unique ranking has been assigned to a sufficient number of available candidate local regions, in order to replace those middle level target parts, by the

candidates with high rank, which were removed due to being identified as drifting.

4.3.2. Updating old target parts

For those middle level target parts that were matched strongly onto candidates in the new frame, the feature models are updated according to new observations. Note that any kind of target model relearning is potentially dangerous, since even small tracking errors can easily cause background pixels to be learned into the target model, leading to instability with exponentially increasing errors. Early colour particle filter work [27], and recent state-of-the-art patch-based methods [35], perform model relearning at a fixed update speed. In contrast, we continuously recompute individual update speeds for each middle level target part at each frame. Our premise is that parts can be relearned rapidly when there is a high confidence in their matching, whereas the relearning rate should be reduced under conditions of uncertainty. We therefore update each patch, using a continuously relearned parameter, $\mu_k^{i_{mid}(p)}$, according to the following update rule:

$$\boldsymbol{M}_k^{i_{mid}(p)} = (1 - \mu_k^{i_{mid}(p)})\boldsymbol{M}_{k-1}^{i_{mid}(p)} + \mu_k^{i_{mid}(p)}\boldsymbol{M}_{obs}^{i_{mid}(p)} \tag{19}$$

$$\mu_k^{i_{mid}(p)} = \hat{L}(\boldsymbol{M}_{k-1}^{i_{mid}(p)}, \boldsymbol{M}_{obs}^{i_{mid}(p)}) \tag{20}$$

where $\boldsymbol{M}_{k-1}^{i_{mid}(p)}$ is the model of the pth target part, in the ith feature modality, at frame $k-1$, while $\boldsymbol{M}_{obs}^{i_{mid}(p)}$ is the observed model of the corresponding matched super-pixel in the current frame. Again, $\hat{L}(.)$ is a similarity metric for the i^{th} feature modality, as defined in Eq. 5.

4.3.3. Updating the top level target representation

At each frame, once all middle level target parts have been either switched off, updated, or replaced, then the top level target model is updated according to Eq. 8, as described in Sec. 3.

4.4. Handling occlusions

The proposed tracker utilises a memory which memorises the latest tracker state, including all middle level target part models. As described in Sec. 4.2, partial occlusion is handled by temporarily switching off poorly matching middle level parts, but retaining these in memory and reacquiring them once occluded target parts reappear in later video frames. The tracker is regarded as being in

a special state of full occlusion if a large proportion of patches (defined by a threshold parameter) remain unmatched after the matching procedure of Sec. 4.2:

$$\mathbb{O}_k = \begin{cases} 1, N_o/N_s > T_o \\ 0, \ \ others \end{cases} \tag{21}$$

where $\mathbb{O}_k$ is occlusion status (1. occlusion; 0. non-occlusion), N_o is the number of occluded parts and N_s is the total number of all parts. T_o is the threshold to check the full occlusion status. In the full occlusion state, the target model updating (at all model levels) stops, and the top level sample propagation scope $V_k(j)^{\texttt{top}}$ in Eq. 12 is enlarged (twice) to handle an increased degree of uncertainty. After finding the best candidate region from the Eq. 13, the algorithm performs the procedure in Sec. 4.2. The tracker returns to the normal (non-occlusion) state once a sufficient proportion of middle level parts (use the same way in Eq. 21) are matched strongly to candidate local regions (super-pixels).

4.5. Summary of the proposed method

The overall algorithm of the proposed method is summarised in Tab. 1.

5. Experimental results and analysis on conventional RGB image sequences

In this section, we first test the performance of our tracker on sequences from the publicly available datasets VOT 2013, VOT 2014 [2] and CVPR 2013 [39], which together comprise 70 sequences in total. These datasets are currently considered state-of-the-art benchmarks in the 2D visual object tracking community. More details of the datasets can be found on the web-pages of [2] and [1]. Note that the VOT and CVPR benchmarks [2] [39], also represent two different evaluation methodologies. We show that the proposed tracker is more robust than the best state-of-the-art methods from both of those tracking challenges, while also offering competitive tracking precision. Next, we carry out a decomposition analysis to understand which parts of the novel tracker contribute to this strong performance. The key elements of the tracker include i) the clustered decision tree, ii) the use of super-pixels to define middle-level target parts or patches, iii) the use of an adaptive target re-learning rate, which re-learns faster during periods of high confidence, and slows down target re-learning during periods of greater uncertainty. We decompose the performance by evaluating the tracker with and without each of these novel elements. The contribution of the clustered decision tree is evaluated, by replacing it with a more conventional parts matching method

Table 1: The pseudo code of the proposed method

Visual object tracking
Input: A bounding box placed around the target object.
Initialisation: Segment the bounding box using SLIC method [3] and generate middle level parts (Sec. 3.2); Extract top level model from middle level parts (Sec. 3.3); Associate the likelihood with image pixels (Sec. 3.4).
For Frame = 2: N_{frame}
Tracking: Generate candidate samples, Eq. 12, and estimate initial target position, Eq. 14 (Sec. 4.1).
Segment (estimated) target region and extract middle level parts (Sec. 4.2)
For id = 1: N_{parts}
While unused feature > 1
Cluster the candidate superpixels, Eq. 15.
Match local parts with clusters/superpixels and estimate possible occlusion, Eq. 16.
End
End
If $N_o/N_s > T_o$ (Eq. 21 in Sec. 4.4)
Adaptation: Generate new parts (Sec. 4.3.1); Update old parts using Eq. 19 (Sec. 4.3.2); Update top level model accordingly (Sec. 4.3.3).
Else
Enlarge the searching scope.
End
End
Output: The estimated object position and scale.

from the literature and comparing performance. The contribution of super-pixels for determining mid-level target parts is evaluated by replacing it with a more conventional method from the literature (random assignments of target regions to be mid-level parts). The contribution of the variable adaptive re-learning rate is evaluated by replacing it with a fixed (mid-range) re-learning rate (as in most of the comparable literature). In each case, all other parts of the algorithm are kept constant. Furthermore, we also investigate how the noisy initialization conditions affect the tracking performance and show that the influence of noisy initialization is within an acceptable range.

5.1. Implementation

The proposed adaptive clustered decision tree structure is designed to handle, in principle, arbitrarily many features in a robust and efficient manner. For proof of principle, we demonstrate the performance of the method on 2D RGB images by implementing a tree with just two features, however this already delivers competitive performance on benchmark test data.

For our primary (top of the tree) feature in the middle level parts matching, we use simple pixel RGB colour values, i.e. $\mathbf{f}_k^{1(x)}$ returns the RGB value of pixel $\mathbf{I}_k(x)$. For the primary feature model, $\boldsymbol{M}_k^{1(p)}$, we use a simple but powerful colour histogram [27]:

$$\boldsymbol{M}_k^{1(p)}(b) = \sum_{x \in R_k^p} \delta[b\mathbf{f}_k^{1(x)}]/N_{pixel} \tag{22}$$

where δ is the Kronecker delta function. b is the bin in the feature model and N_{pixel} the number of the pixels inside the region R_k^p.

For this colour feature, we use a likelihood function between two candidate image regions, $\hat{L}_k^{1(p,q)}$, derived from the Bhattacharyya metric [5]:

$$\hat{L}_k^{1(p,q)} = \sum_{b=1}^{N_b} \sqrt{\boldsymbol{M}_k^{1(p)}(b) * \boldsymbol{M}_k^{1(q)}(b)} \tag{23}$$

where N_b is the number of the bin.

For the secondary feature, we employ a motion feature, where candidate local regions are assigned high matching likelihoods if they employ a small frame-to-frame motion for the part being matched, which can be denoted as:

$$\hat{L}_k^{2(p,q_t)} = \| R_k^p - R_k^{q_t} \| \tag{24}$$

where R_k^p is the image coordinates of the candidate local region p while $R_k^{q_t}$ is the image coordinates of target part q_t.

In the model updating (in Sec. 4.3.1), to form the 1^{th} level of the trees for drifting parts replacement, we use the 1^{th} feature modality to calculate, for every unmatched candidate, the primary expected likelihood $\mathbf{S}_k^{1(p_s)}$ (in Eq. 18) of that candidate's constituent pixels belonging to the target, which is computed as:

$$\mathbf{S}_k^{1(p_s)} = \frac{1}{N_{p_s}} \sum_{x \in R_k^{p_s}} l_k^{1(x,\mathtt{bot})} \tag{25}$$

Table 2: Values of key algorithmic parameters

Section	Equation	Value
Initialization	$\lambda^{\mathtt{bot}}$ in Eq. 11	0.1
Decision tree	λ_{rank} in Eq. 18	0.1
Occlusion	T_o in Eq.21	40%

where $\mathbf{S}_k^{1(p_s)}$ is the expected likelihood of the p_sth candidate local region belonging to the target, N_{p_s} is the number of pixels in the p_sth candidate, and $l_k^{1(x,\mathtt{bot})}$ is as defined in Eq. 11.

For the secondary feature for updating, we utilize a dense feature to assign an additional likelihood to the clustered candidate local regions:

$$\mathbf{S}_k^{2(p_s)} = eps(-\|R_k^{p_s} - R_k^c\|) \tag{26}$$

where $R_k^{p_s}$ is the candidate local region p_s while R_k^c is the centre region of the matched parts.

The tracking algorithm has been implemented on an Intel Core i5-3570 CPU, using Matlab code (linked also to some C++ components). This unoptimised implementation achieves near-to-real-time performance of 8fps. The key parameters initialized in the first frame are listed in the Tab. 2.

5.2. VOT tracking challenge and its evaluation protocol

We first evaluate our tracker using the ICCV2013 [39] and ECCV2014 [2] "VOT challenge" testbeds, which have widely been used as a standard tracking evaluation protocol. Since our tracker is a deterministic algorithm (always generates identical results), one pass evaluation is adopted in the protocol. Whenever a tracking failure is detected (i.e., the bounding box has zero overlap with ground truth), the tracker is re-initialised. Tab. 3 only shows the top five evaluated VOT trackers, out of around 30 that were evaluated by the challenges. Following VOT challenge evaluation protocol, "Fail." shows the total number of failure instances, while "Acc." indicates the average overlap between each tracker's output bounding box and the ground truth bounding box.

Our tracker has zero failures in VOT2013 and only one failure in VOT2014. The next best algorithm is PLT which also achieved zero failures in VOT2013. Note that the version of PLT tested in VOT2013 used a fixed bounding box size. Therefore this algorithm was unable to adapt to targets which change their size during tracking (e.g. due to range changes). However, since most objects in most

Table 3: VOT challenge results: comparing against best 4 trackers

	VOT 2013 (16 sequences)				
	Ours	PLT13 [15]	LGT++ [41]	EDFT [10]	FoT [36]
Fail.	0	0	1.53	14	22
Acc.	0.59	0.58	0.57	0.58	0.63
	VOT 2014 (25 sequences)				
	Ours	PLT14 [15]	DGT [7]	DSST [9]	SAMF [21]
Fail.	1	4	25	29	32
Acc.	0.52	0.56	0.58	0.62	0.61

test sequences luckily stayed roughly the same size in the VOT2013 benchmark videos, this rigid size constraint helped the algorithm to achieve an (arguably artificially) high robustness score. For VOT2014, a different version of PLT was submitted, which did enable adaptation to changing target size but failed more.

5.3. CVPR tracking challenge and its evaluation protocol

For a more extensive comparison, we also combine the VOT test sequences with all those from the CVPR 2013 tracking benchmark data set [39]. Using this 70-sequence dataset, we compared our method against the publicly available trackers which have showed strong performance in either the VOT or the CVPR tracking challenges, namely: Struck [14], SCM [43], LGT++ [41], CSK [16], IVT [30], L1 [25], and PF [27]. Note that this experiment is complementary to the previous VOT testing. In particular, since this dataset contains instances of full occlusions, the evaluation is conducted without re-initialization after tracking failures. We show the results as trade-off curves in terms of overlap, recall and precision. The success ratio (y-axis) indicates the number of frames with higher (overlap/recall/precision) score than a defined threshold (x-axis).

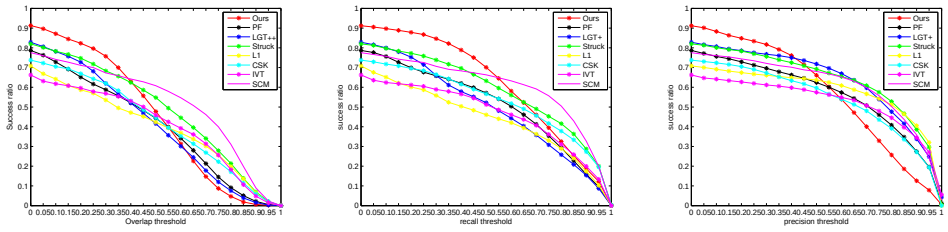

Figure 6: The success ratio versus overlap threshold, recall threshold and precision threshold curves in 70 sequences.

As shown in Fig. 6, our tracker achieves better success ratio in overlap ranges up to 0.4 in the overlap trade-off curve. Similar performance could be observed

in recall and precision curves. Note that around 10% of the superpixels only use the first feature. Besides the strong performance, the computation cost is also reduced.

Every sequence used by the benchmark has been annotated with attributes, i.e., deformation, illumination, and occlusion, which explicitly highlight the target tracking performance in such conditions. To further evaluate the performance when handling various severe tracking conditions, we also show the trade-off curves for those test videos identified in the benchmark challenges as containing the attributes of: significant target deformations, Fig. 7; severe illumination changes, Fig. 8; and occlusions, Fig. 9.

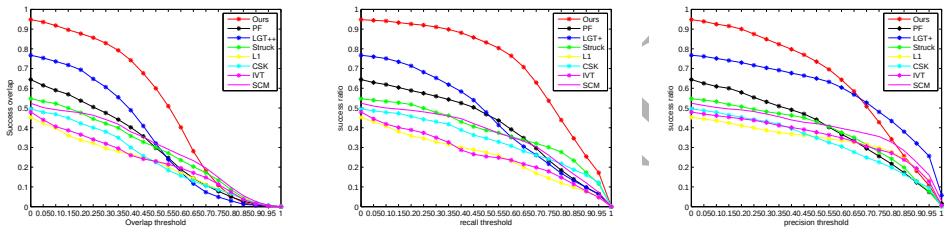

Figure 7: The success ratio versus overlap threshold, recall threshold and precision threshold curves in 19 sequences with deformation.

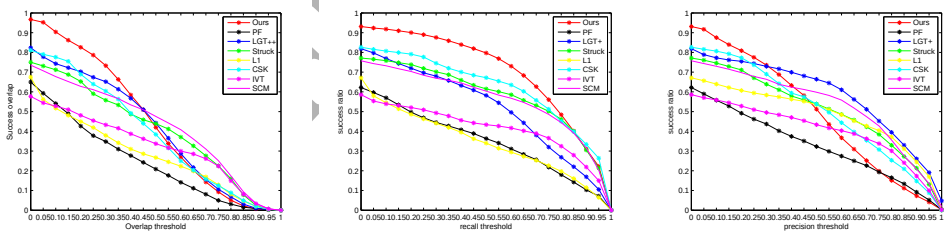

Figure 8: The success ratio versus overlap threshold, recall threshold and precision curves in 18 sequences with illumination change.

From the trade-off curves in Fig. 7, one can see that our tracker outperforms the other methods in cases of highly deformed targets. We attribute this performance to the flexibility of the clustered decision tree approach to online model relearning. The tracker also achieves competitive results in illumination-change (Fig. 8) and occlusion scenarios (Fig. 9). We attribute the strong performance under illumination changes to the robustness of the cross-constrained multi-level

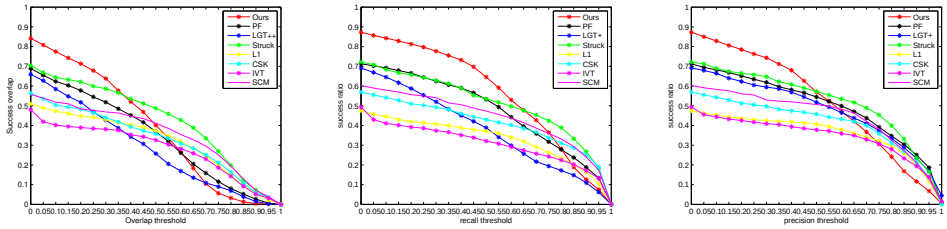

Figure 9: The success ratio versus overlap threshold curve, recall threshold and precision curves in 22 sequences with occlusion.

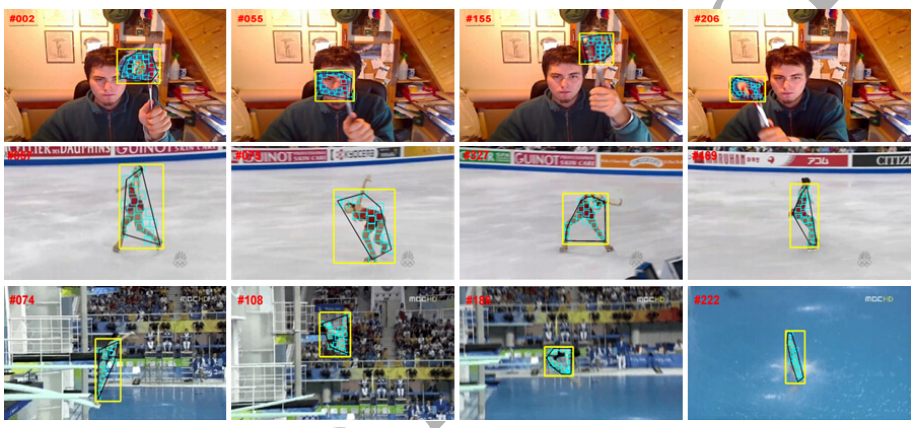

Figure 10: Visualization of results on sequences: *Torus*, *Iceskating*, *Diving*, showing extreme target deformations and significant clutter.

target model. We attribute the results of the occlusion tests to the generality and adaptability of the proposed method. Note that the proposed tracker shows better performance in terms of recall compared to precision, which shows that our tracker can cover most parts of the true target. To have a more intuitive understanding of the tracker, handling various tracking challenges, we also visualise some examples in Fig. 10, which feature extremely deforming targets (e.g., a skater) and very strong background clutter (e.g., a diver).

5.4. Performance contributions of tracker sub-components

The previous section has shown that our proposed tracker performs extremely robustly in comparison to the best state-of-the-art trackers from the literature. However, our tracker relies on several key components which differ from other tracking methods. This section investigates the extent to which each of these com-

ponents contributes to the overall strong performance of the tracker. To demonstrate the effect of each novel component, we evaluate performance while replacing that component with a conventional method from the literature, but keeping all other parts of the algorithm the same. Firstly, we remove our proposed use of super-pixels for determining mid-level target parts, and replace it with the well-known method of [35], which selects random target regions to be mid-level patches. Secondly, we remove our proposed adaptive clustered decision trees (which adaptively combine different features according to their importance) with a standard homogeneous feature fusion method, as in [35] and much of the feature fusion literature, wherein the likelihoods of different features are simply multiplied. Thirdly, we remove our proposed adaptive model re-learning speed, and replace it with a fixed re-learning speed of 0.5 in Eq. 20. Fig. 11 compares the performance of the complete proposed tracker, with performance of the tracker when each sub-component is removed and replaced by a conventional method from the literature.

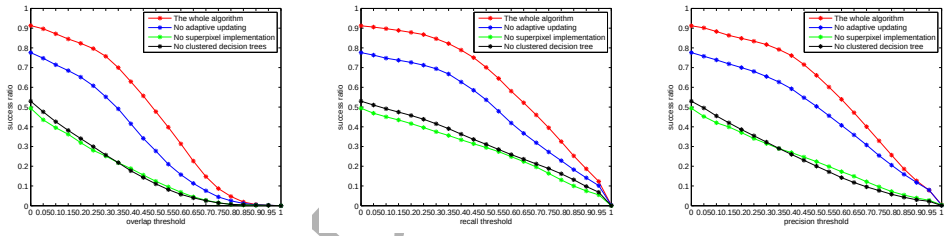

Figure 11: Success ratio versus overlap, recall and precision threshold curves for 70 sequences. Comparison of proposed tracker versus modified tracker with each of three key components replaced by conventional methods from the literature.

Fig. 6 suggests that the adaptive re-learning speed does have a significant impact on performance. However, even without the adaptive re-learning speed, the tracker is still able to deliver comparable performance to the state-of-the-art trackers analysed in Fig. 6. In contrast, removing either the super-pixels mid-level parts representation or the clustered decision tree matching method, causes a larger degradation in performance. It appears that the superpixels representation and the clustered decision tree method contribute equally to the overall performance of the tracker, and that their contribution is more important than the effect (still significant) of the adaptive re-learning rate.

As discussed also in earlier sections, we suggest the following possible reasons for the effects of these techniques on overall performance. Firstly, because

superpixel segmentation results in homogeneous image regions, it is more likely to lead to mid-level patches which contain purely foreground pixels or purely background pixels. Target patches which erroneously contain background pixels will then be rapidly eliminated from the target model by other parts of the algorithm, leaving mid-level patches which reliably adhere only to the target pixels, and this supports robust and stable tracking. Additionally, without super-pixel segmentation, the matching routine must exhaustively search all possible local candidate regions, as in [35], to achieve a robust match. One strong benefit of the clustered decision tree, is that it adaptively weights in favour of whichever features are most discriminating for any particular patch, at any given frame. In contrast, conventional feature fusion (by multiplying feature likelihoods) enables a poorly performing feature (e.g. where the background is similar to the foreground in that feature modality) to damage the good performance which might be achieved by another feature that happens to be more discriminating in the current frame.

5.5. Robustness evaluation with noisy initialization

We also investigate how the initialization affects the tracking performance in Fig. 12. Instead of using the ground-truthed bounding box in the first frame, we perturb the bounding box positions with 5% and 10% distance shift in each of four directions, i.e. left, right, up, down. It shows that the tracking performance decreases slightly with noisy conditions, while still delivers competitive results compared to other trackers.

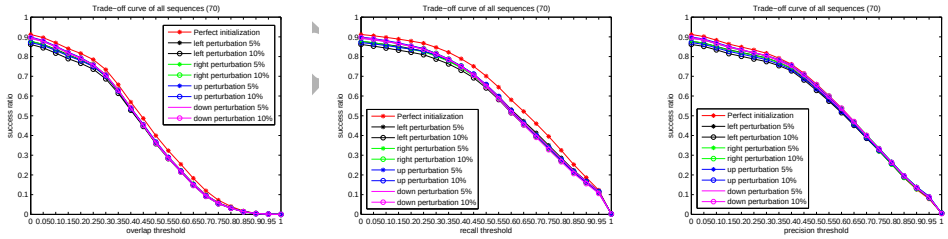

Figure 12: Success ratio versus overlap, recall and precision threshold curves for 70 sequences with various initialisation conditions.

6. Extensions of clustered decision tree tracking to RGB-D data

In this section, we illustrate the flexibility and generality of the clustered decision tree approach, by showing how the tree can easily be extended to include arbitrary kinds of new features, i.e. RGB-D data. We evaluate the resulting RGB-D

tracker using the public benchmark data-set of [32], demonstrating the flexibility of the proposed clustered decision tree structure to handle arbitrary numbers and kinds of features. We explore the extent to which the performance of the proposed clustered decision tree is dependent on the order in which it handles different features. We evaluate the tracker using three features (colour, motion and depth), while testing all six possible permutations of feature order. We show that overall performance remains very similar for all feature orders.

6.1. Dynamic feature selection capability of clustered decision trees

We first show how the two-level tree, evaluated in the previous section using RGB images, can be extended to a three-level tree, where the third level encodes a third feature of depth, obtained from RGB-D images. For proof of principle, we embed the depth information by using its absolute value. When the tree grows as far as the depth feature level, the parts matching is conducted by selecting the candidate local regions with the least movement in depth, compared to the corresponding target parts. The clustering is performed according to standard deviation. We demonstrate the RGB-D extension of our algorithm using 5 publicly available RGB-D sequences [32] (for which groundtruth annotation is available).

To illustrate the effect of each feature (and each level of the decision tree), we compare the performance of the tracker using one feature/tree-level, two features/tree-levels and all three features/tree-levels. The performance of the tracker with different numbers of features/tree-levels is compared in Fig. 13, where C represents the colour feature, M represents the motion feature and D denotes the depth feature. We can see that, adding a second feature (tree level), improves the performance significantly. Adding a third feature (tree level) makes further improvement. Note that, all parts matching operations, at all frames, make use of the primary feature (colour). For 78% of the operations, the clustered decision tree detects that the colour feature alone does not provide sufficient confidence, and so progresses to the second level of the tree where it exploits the secondary feature (motion). In 33% of the matching operations, the clustered decision tree additionally progresses to the third tree level, to make use of the additional depth feature. The reason that the third feature contributes less improvement than the second feature, is that the algorithm makes use of the second feature much more frequently than the third feature.

A key strength of the clustered decision tree is that it automatically and dynamically selects only those features which are necessary to achieve confident matching of each mid-level target part at each frame. Conversely, compared to the conventional all-feature-fusion approaches, significant efficiency is gained by

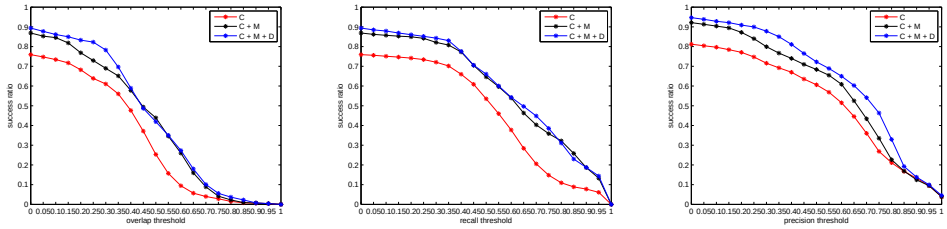

Figure 13: Comparison of the performance of trackers which make use of one, two and three-level clustered decision trees, where the successive tree levels exploit colour, motion and depth features respectively. Graphs show the success ratio versus overlap threshold curve for each tracker, when tested on five RGB-D sequences from a public benchmark data set. C, M, D denote colour, motion and depth features respectively.

reducing the use of the features, e.g., by 22% the second feature and 67% the third feature in the whole dataset.

6.2. *Effect on performance of the order of features*

The clustered decision tree steps through features in a particular order, with each feature being assigned to a particular level of the tree. Each feature must be manually assigned to a tree level, i.e. a feature order must be specified, and it is not obvious how to choose this feature order. In this section, we evaluate the effects on performance of adopting a variety of different feature orders, and show that the algorithm performance is reasonably indifferent to the choice of order.

In the case of a three-level tree, processing colour, motion and depth features, there are six possible permutations of feature order in the tree. Fig. 14 graphs the performance curves of all six permutations, when testing the proposed tracker on five ground-truthed RGB-D sequences drawn from [32] benchmark data set.

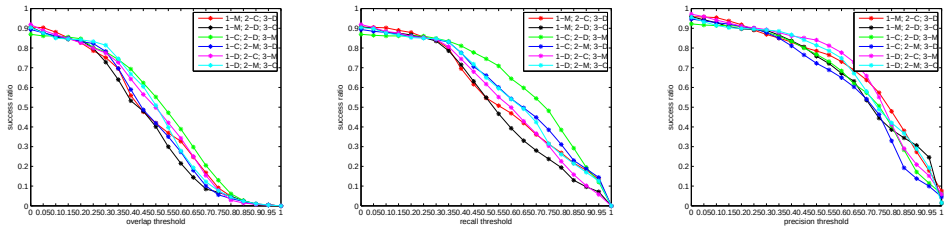

Figure 14: The success ratio versus overlap threshold, recall and precision curves for all six possible features orders in RGB-D sequences. C, M, D denote colour, motion and depth features respectively.

Fig. 14 suggest that the order of features in the tree does not greatly affect overall tracking performance. It is interesting to notice that, when the color and motion features occur consecutively, then swapping their order makes almost no difference. In contrast, when color and motion occur either side of the depth feature, then swapping the order of colour and depth makes a greater difference. Possibly this might be due to our somewhat naive and simplistic use of the depth feature in this work (simply using raw depth values as a feature). An even greater robustness to feature order might result from features that were equally well utilised, with equally well chosen clustering and likelihood functions. Note that our intention here was not to present a novel and clever use of depth features, but merely to illustrate how generally and flexibly the clustered decision tree concept can be extended to include new features of arbitrary kinds.

7. Conclusion

This paper has presented a novel visual object tracking algorithm, which demonstrates extremely robust performance compared to state-of-the-art methods from the recent literature. The proposed algorithm outperforms the best algorithms from each of the VOT2013 and VOT2014 benchmark tracking challenges, and outperforms 7 state-of-the-art trackers on the CVPR2013 benchmark tracking data set. The tracker is especially robust against challenging tracking conditions of large target deformation, rapid illumination changes, and occlusions.

The key components of the proposed tracker are: 1) a multi-level target representation, where targets are modelled at three different levels of granularity (pixel level, part level and bounding box level), which are cross-constrained to enable robust model relearning; 2) the use of superpixel segmentation to determine which target parts to select as middle-level parts; 3) an adaptive clustered decision tree method which dynamically selects the minimum combination of features necessary to sufficiently represent each target part at each frame, thereby providing robustness with computational efficiency; 4) the use of such adaptive trees to, firstly, perform matching at the parts level and, secondly, to select the best candidates for learning new parts of the target when old parts become obsolete or occluded; 5) the use of an adaptive model re-learning rate which improves stability by reducing model re-learning during periods of low confidence.

To explain the strong performance of the algorithm, we have decomposed it by testing with each of the key components switched off and replaced by a conventional method from the literature, while preserving all other components. This analysis suggests that the use of superpixels to determine middle-level target parts,

and the clustered decision tree parts matching method, contribute the most (and contribute equally) to performance. The use of the adaptive target re-learning rate makes a smaller but still significant contribution to overall performance. In addition, we also evaluate the tracking performance with noisy initialization conditions, where the tracker still delivers competitive performance under such conditions.

To explore the generality and flexibility of the adaptive clustered decision tree structure, we have shown how it can easily be extended to other tracking scenarios. As an example, we showed how to extend the tree to include a depth feature from RGB-D image sequences. We also investigated the contributions of each level of the clustered decision tree to the overall performance, and we illustrated how the clustered decision tree efficiently, adaptively and dynamically varies the number of features exploited for matching each target part at each frame, using a single feature for easy parts/frames and multiple features for difficult parts/frames. Finally, we empirically analysed the effect of choosing different permutations for which feature is allocated to which level of the decision tree. We concluded that the order of features in the tree makes little difference to the overall tracking performance.

In our future work we plan to extend the proposed single-target tracker to multi-target scenarios, for example in team-sports videos [42], which require additional layers (possibly also encoded as decision trees) to capture interactions between multiple targets (players) and global game strategy.

Acknowledgement

This work was funded by EU H2020 RoMaNS 645582 and EPSRC EP/M026477/1. We also acknowledge MoD/Dstl and EPSRC for providing a Department of Defence funded MURI project which supported the involvement of Aleš Leonardis. The National Key Research and Development Program of China, No.2016YFC0103100.

References

[1] The CVPR benchmark dataset. `https://sites.google.com/site/trackerbenchmark/benchmarks/v10`.

[2] The VOT challenge. `http://www.votchallenge.net/`.

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012.

[4] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *PAMI*, 33(8):1619–1632, 2011.

[5] Anil Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhyā:The Indian Journal of Statistics*, pages 401–406, 1946.

[6] Paul Brasnett, Lyudmila Mihaylova, David Bull, and Nishan Canagarajah. Sequential monte carlo tracking by fusing multiple cues in video sequences. *Image and Vision Computing*, 25(8):1217–1227, 2007.

[7] Zhaowei Cai, Longyin Wen, Jianwei Yang, Zhen Lei, and Stan Z Li. Structured visual tracking with dynamic graph. In *ACCV*, pages 86–97. Springer, 2013.

[8] Robert T Collins, Yanxi Liu, and Marius Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005.

[9] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[10] Michael Felsberg. Enhanced distribution field tracking using channel representations. In *ICCV visual object tracking workshop*, pages 121–128. IEEE, 2013.

[11] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[12] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *CVPR*, volume 1, pages 260–267. IEEE, 2006.

[13] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, pages 234–247. Springer, 2008.

[14] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270. IEEE, 2011.

[15] Cher Keng Heng, Sumio Yokomitsu, Yuichi Matsumoto, and Hajime Tamura. Shrink boost for selecting multi-lbp histogram features in object detection. In *CVPR*, pages 3250–3257. IEEE, 2012.

[16] Joao F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, pages 702–715. Springer, 2012.

[17] Seunghoon Hong and Bohyung Han. Visual tracking by sampling tree-structured graphical models. In *ECCV*, pages 1–16. Springer, 2014.

[18] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, pages 1822–1829. IEEE, 2012.

[19] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[20] Junseok Kwon and Kyoung Mu Lee. Highly nonrigid object tracking via patch-based dynamic appearance modeling. *PAMI*, 35(10):2427–2441, 2013.

[21] Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV visual object tracking workshop*, pages 254–265. 2014.

[22] Haibin Ling and Kazunori Okada. Diffusion distance for histogram comparison. In *CVPR*, volume 1, pages 246–253. IEEE, 2006.

[23] Yang Lu, Tianfu Wu, and Song-Chun Zhu. Online object tracking, learning and parsing with and-or graphs. In *CVPR*, pages 3462–3469. IEEE, 2014.

[24] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *PAMI*, 26(6):810–815, 2004.

[25] Xue Mei and Haibin Ling. Robust visual tracking using L1 minimization. In *ICCV*, pages 1436–1443. IEEE, 2009.

[26] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, volume 2, pages 2161–2168. IEEE, 2006.

[27] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. An adaptive color-based particle filter. *Image and vision computing*, 21(1):99–110, 2003.

[28] Toufiq Parag, Fatih Porikli, and Ahmed Elgammal. Boosting adaptive linear weak classifiers for online learning and tracking. In *CVPR*, pages 1–8. IEEE, 2008.

[29] Federico Pernici and Alberto Del Bimbo. Object tracking by oversampling local features. *PAMI*, 36(12):2538–2551, 2013.

[30] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[31] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *IJCV*, 40(2):99–121, 2000.

[32] Shuran Song and Jianxiong Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In *ICCV*, pages 233–240, 2013.

[33] Rustam Stolkin and Mohammed Talha. Particle filter tracking of camouflaged targets by adaptive fusion of thermal and visible spectra camera data. *IEEE Sensors*, 14(1):159–166, 2014.

[34] Zhuowen Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *ICCV*, volume 2, pages 1589–1596. IEEE, 2005.

[35] Luka Čehovin, Matej Kristan, and Aleš Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *PAMI*, 35(4):941–953, 2013.

[36] Tomáš Vojíř and Jiří Matas. Robustifying the flock of trackers. In *16th Computer Vision Winter Workshop*, pages 91–97, 2011.

[37] Qing Wang, Feng Chen, Wenli Xu, and Ming-Hsuan Yang. Online discriminative object tracking with local sparse representation. In *WACV*, pages 425–432. IEEE, 2012.

[38] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *ICCV*, pages 1323–1330. IEEE, 2011.

[39] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, pages 2411–2418. IEEE, 2013.

[40] Jingjing Xiao, Linbo Qiao, Rustam Stolkin, and Aleš Leonardis. Distractor-supported single target tracking in extremely cluttered scenes. In *ECCV*, 2016.

[41] Jingjing Xiao, Rustam Stolkin, and Aleš Leonardis. An enhanced adaptive coupled-layer lgtracker++. In *ICCV visual object tracking workshop*, volume 2, pages 1–8. 2013.

[42] Jingjing Xiao, Rustam Stolkin, and Ales Leonardis. Multi-target tracking in team-sports videos via multi-level context-conditioned latent behaviour models. *BMVC*, 2014.

[43] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, pages 1838–1845. IEEE, 2012.

Jingjing Xiao received her Bachelor and Master degree from College of Mechatronics Engineering and Automation, National University of Defence Technology, China, in 2010 and 2012, respectively. She holds a PhD degree from University of Birmingham in 2016. Currently, she is a lecturer in Third Military Medical University, China, and an hononary research fellow in the University of Birmingham, U.K.. Her research interests include single object tracking, multi-object tracking with computer vision.

Rustam Stolkin currently serve as the Senior Birmingham Fellow in Robotics, based in the Department of Mechanical Engineering at University of Birmingham, where he has worked since 2008. His training included undergraduate and masters degrees in Engineering from Oxford University, and a PhD in Robot Vision undertaken between University College London and UK imaging industry. He also worked as an Assistant Professor (Research) at Stevens Institute of Technology, USA, 2004-2008. His main interests include vision and sensing, robotic grasping and manipulation, robotic vehicles, human-robot interaction, AI and machine learning.

Ales Leonardis is a Professor at the School of Computer Science, University of Birmingham and co-Director of the Centre for Computational Neuroscience and Cognitive Robotics. He is also a Professor at the FCIS, University of Ljubljana and adjunct professor at the FCS, TU-Graz. His research interests include robust and adaptive methods for computer vision, object and scene recognition and categorization, statistical visual learning, 3D object modeling, and biologically motivated vision.